One Engineering System …is it enough?

Galen Hunt

Cross-Division Architect, OSG Apex

Partner Research Manager, Microsoft Research Technologies

http://aka.ms/engsys

Act 1: Tools.

What is an Engineering System?

- The end-to-end **tools** and **processes** for turning source code into deployed products and services
  - "I've turned an idea into a line of source code…"
  - "… how do I **submit** it?"
  - "… how do I **build** it?"
  - "… how do I **test** it?"
  - "… how do I **flight** it?
  - "… how do I **deploy** it?"
  - "… how do I **service** it?"
  - "… how do I **refactor** it?
  - "… how do I **reuse** it in another product?"
  - "… how do I **understand** it in context?"

What is *One Engineering System*?

- A **combined effort** between OSG, DevDiv, and MSR
  to incubate a company-wide engineering system
  by merging 1st party and 3rd party tool efforts.
- *Apex* includes these *north star* **tools** in *Visual Studio Online* (VSO):
  1. new **source** code management strategy (Git)
  2. new **quality** strategy (dev-centric test w/ *inner loop*)
  3. new **build** language and build system (Domino)
  4. **source partitioning** system with build enforcement (SP)
  5. **binary component** system for sharing built artifacts (Nuget)
- EDN0 describes the process we use to evolve these choices…

How can we test for Apex success?

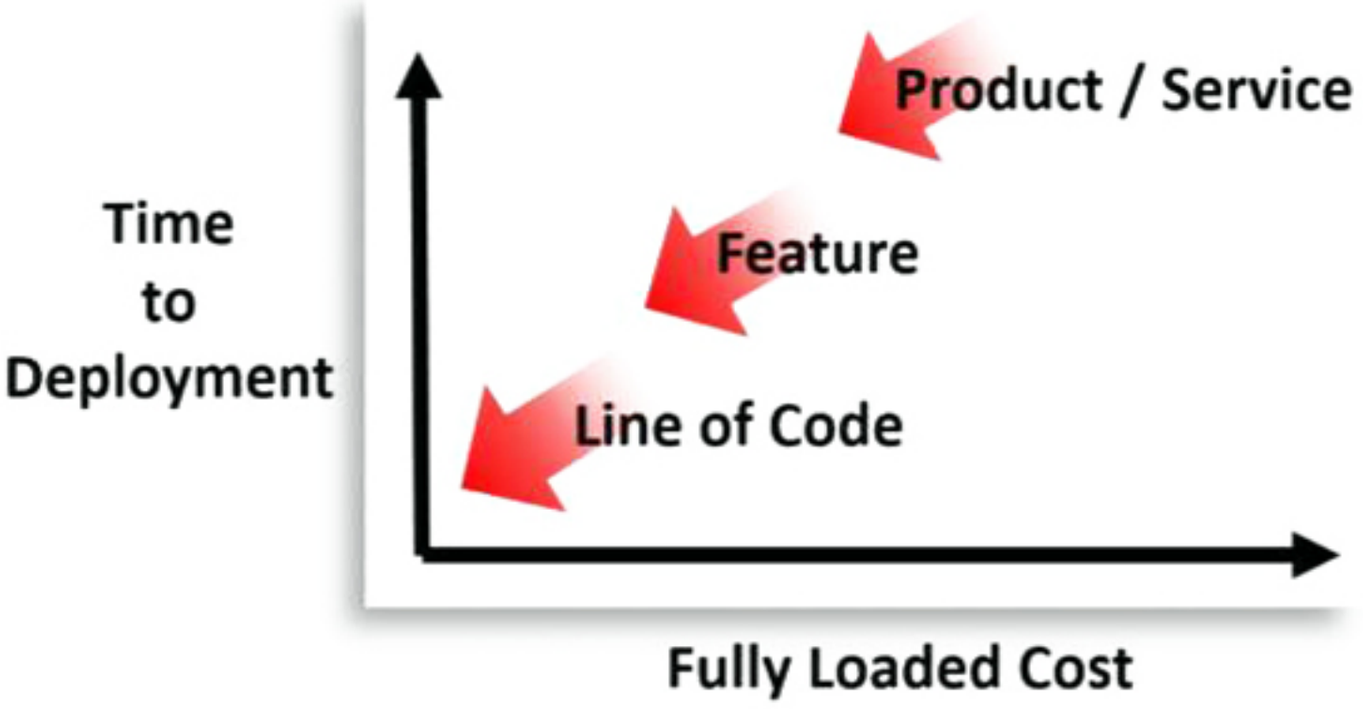
- Can I **find** and **enlist** in the source for any component in the company w/o sending an email*?
- Can I **change** a line of code, **build** the component, **verify** that it passes all RI gates, and **flight** it w/o sending an email*?
- Can I **commit** my changes to a discoverable branch, **submit** a pull request to the owners, and expect them to **accept** it?
- Can I **include** another team's component, **with servicing,** into my product?

* "sending an email" == "requiring help or approval from a human"

Act 2: Why?

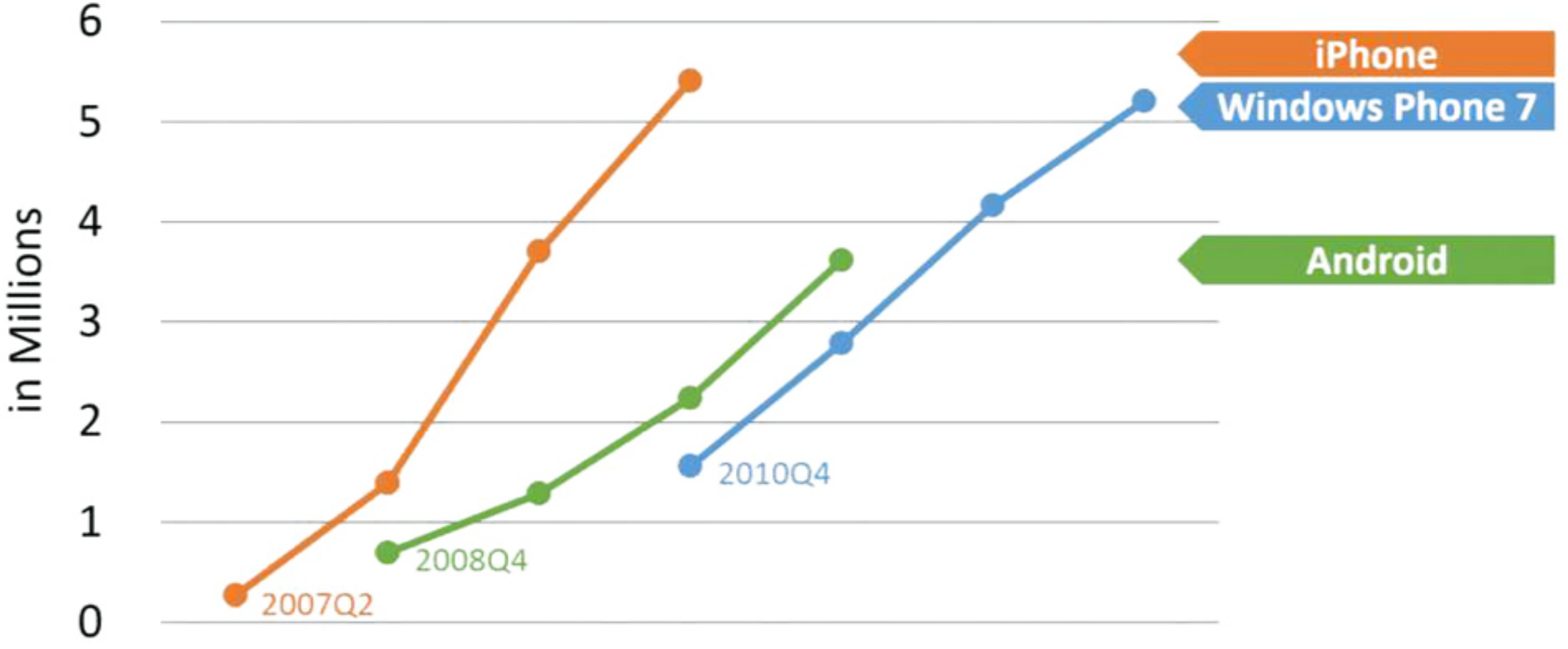
What does the Business need from Apex?

- Allow *business models* to turn source code into money
- Deliver more value with more speed to more customers



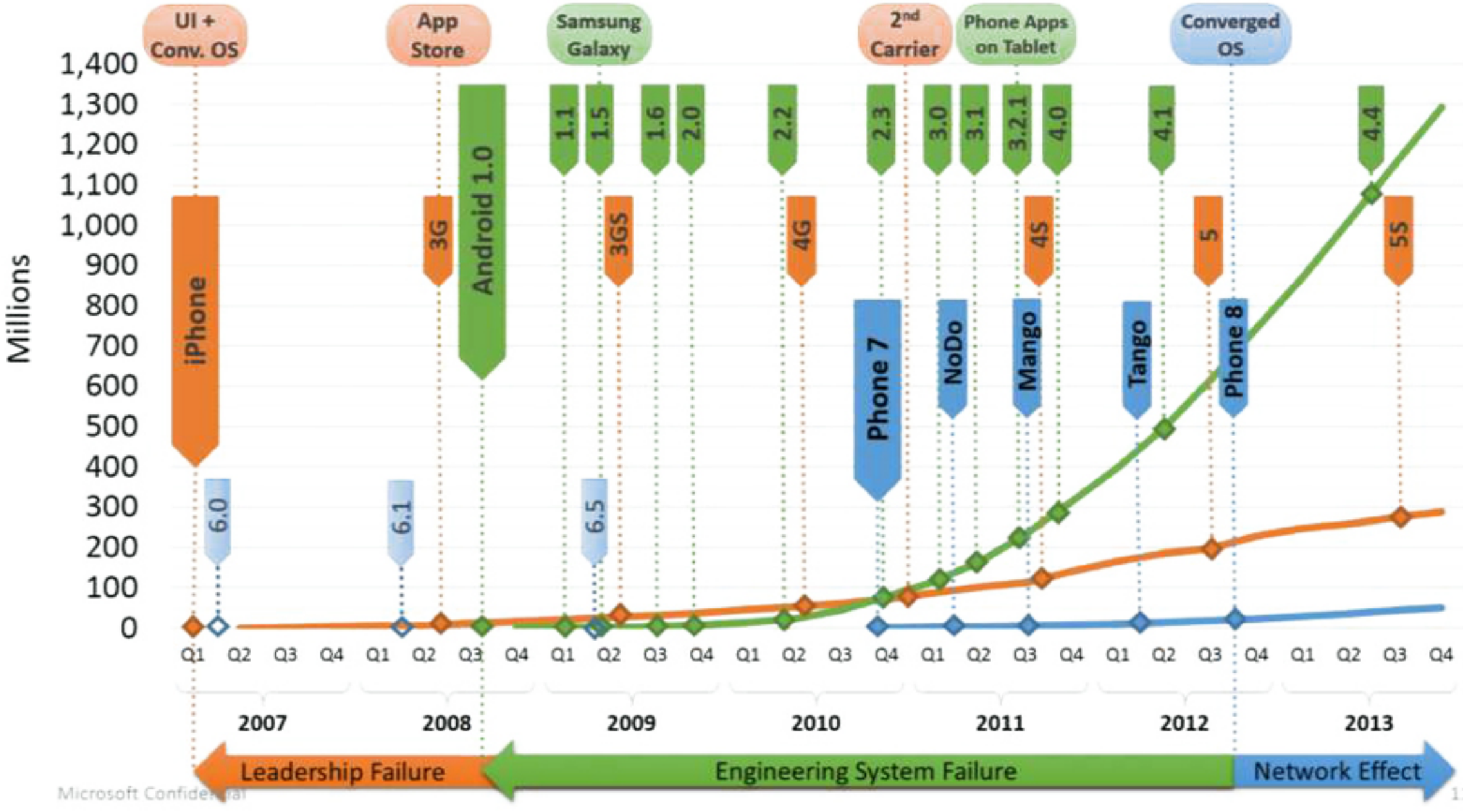
Smartphone Unit Volume

First Year– Plot of Running Totals of Units Sold



* Data Source: IDC Mobile Tracker 2011

Smartphone Unit Volume through 2013 (2YRT)



* Data Sources: IDC Mobile Tracker 2013, Wikipedia

Other product areas for concern…

- **Tablets**
- **Internet Gateways** (aka Wireless routers)
- **SmartTVs**
- **Wearables**
- In-flight entertainment
- In-dash navigation, etc.
- Robotics
- **Networked-connected sensors and devices** (IoT)
- **Cloud appliances**

Act 3: Cultural Change.

1950 INDIANAPOLIS 500

Culture is defined by tools **and** ~~rituals~~ process

67 seconds vs. 3 seconds…

Tools improvements:

- ❑ pneumatic wrench
- ❑ pneumatic jack
- ❑ plexiglas visor

} 2x

Process improvements… } 10x

We need to become more

responsive **[strategy]**

streamlined **[process]**

componentized **[code]**

Responsive*

- ***Autonomous*** teams and individuals
  experiment continuously to learn and remove blind spots
  with ***transparency***.
- If we don't use information, our competitors will.

…being first or loudest does not prove I'm smart.

*Responsiveness vs. Efficient from Adam Pisoni, CTO Yammer

Streamlined

- **Computers can free humans**
  for more meaningful pursuits.
  - people are expensive ($250K/year); computers are not ($2.5K/year)

…inserting a human does not make a process faster or more accurate.

- Fully loaded human cost of…
  - 1-hour build branch delay . . . . . $6,250 (50 x 1 hr)
  - 1-minute delay in “sd sync” . . . . . $2,000,000 (5,000 x 200 min) / yr
  - source access by human approval . . . . . $35,000,000 (35,000 x 8 hr) / yr
  - a 3-month Windows planning process . . . $300,000,000 (5,000 x 480 hrs)

Componentized

- **Our persistent value** is in reusable components.
- We can **manage complexity**, **enable reuse**, & **achieve scale**
  by producing code in discrete units of functionality
  with well-defined interfaces.

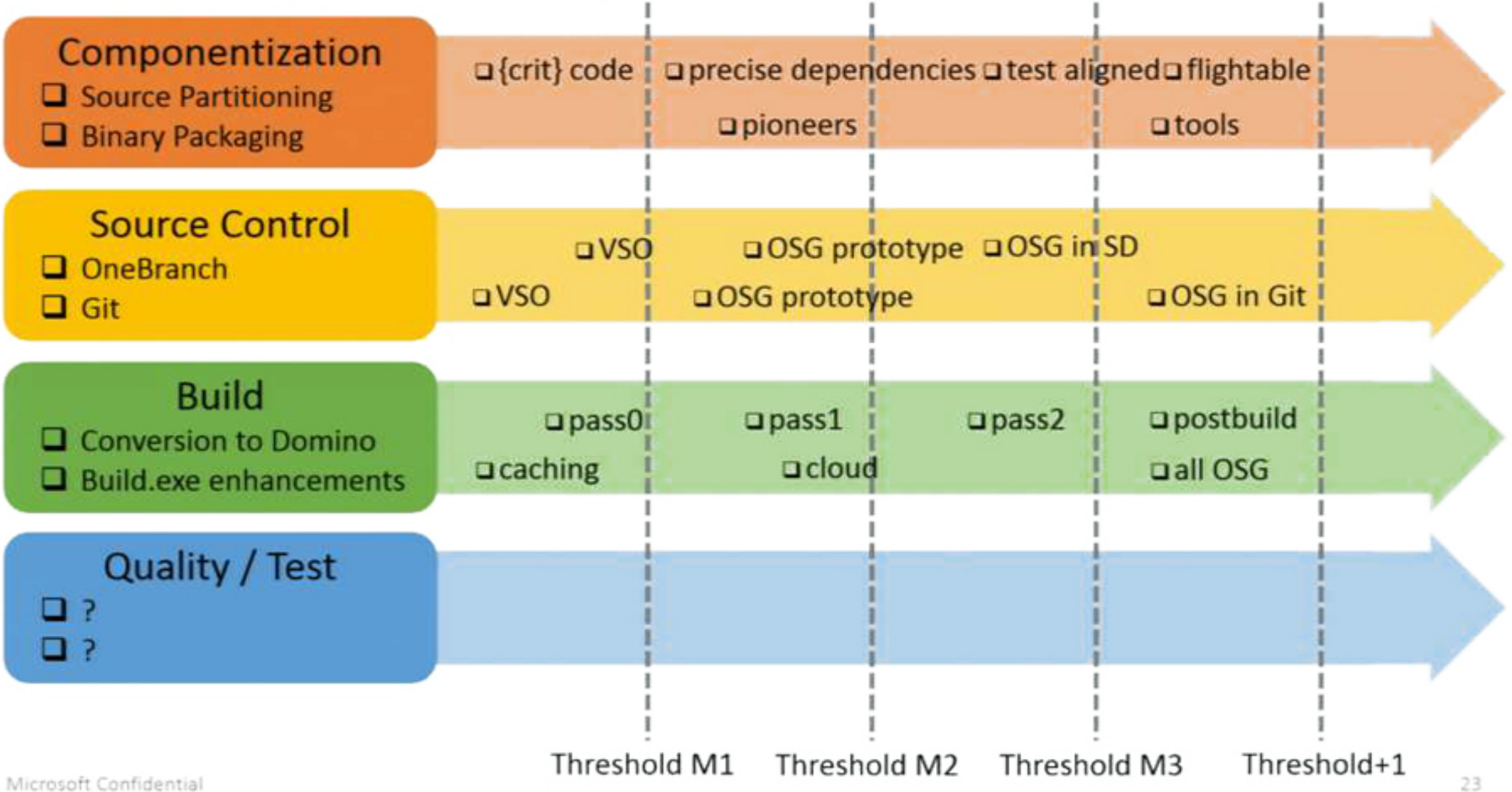
…monolithic, end-to-end solutions do not pivot easily.

- Good components have
  - *precise interfaces* and *private implementations* to enable replacement
  - *scoped dependencies* to facility reuse
  - *distributive* coding, building, testing, flighting, and deployment

… in an evolving world, with accruing debt, the most dangerous choice is the status quo.

Epilogue.

OSG adoption of Apex



The Road to Autonomy

- Small team (2-7 people) **owns its code** and **progresses on its cadence**
  - loosely coupled, tightly aligned with other teams
  - ships in small and frequent releases (~3-week sprints)
- Their day-to-day work is **entirely in their *partition*** (source & build)
- Their partition is **in its own *git repo***.
- They **vet their dependencies** (upstream, downstream, and *friend*).
- They create ***topic branches* on demand**
- They **RI and *publish* on demand**.
- Anyone can submit pull requests, but **only they can *commit* to their mainline**.

Call to Action and Resources

- Don’t be overwhelmed:
  - You are part of a great team.
  - Your work matters.
  - Our leadership is committed to helping you succeed.

- Be the change:
  - Be responsive. Streamline. Componentize.

- Resources:
  - One Engineering System site: http://aka.ms/engsys
    - [Design Notes] : Understand the plans and the process for making decisions (EDN0)
    - [Sprint Emails] : Track the progress of each effort by sprint
    - [Points of Contact] : Representatives from each division
  - Engineering System Architecture Discussions DL: engsysarch
  - Engineering Design Note News DL: engsysnotes